Multi-Entity Bayesian Networks Without Multi-Tears

Paulo C. G. da Costa and Kathryn B. Laskey

George Mason University
4400 University Drive
Fairfax, VA 22030-4400
[pcosta, klaskey]@gmu.edu

Abstract

An introduction is provided to Multi-Entity Bayesian Networks (MEBN), a logic system that integrates First Order Logic (FOL) with Bayesian probability theory. MEBN extends ordinary Bayesian networks to allow representation of graphical models with repeated sub-structures. Knowledge is encoded as a collection of Bayesian network fragments (MFrags) that can be instantiated and combined to form highly complex situation-specific Bayesian networks. A MEBN theory (MTheory) implicitly represents a joint probability distribution over possibly unbounded numbers of hypotheses, and uses Bayesian learning to refine a knowledge base as observations accrue. MEBN provides a logical foundation for the emerging collection of highly expressive probability-based languages. A running example illustrates the representation and reasoning power of the MEBN formalism.

Introduction

Uncertainty is a ubiquitous feature of the world, and probability theory is a natural candidate to represent uncertain phenomena. Application of probability to artificial intelligence was initially hindered by skepticism about tractability of inference and feasibility of representation. This situation changed dramatically with the introduction of Bayesian networks (BNs) (Lauritzen & Spiegelhalter 1988, Pearl 1988) and their application to diverse areas such as language understanding (Charniak & Goldman 1989a, 1989b), visual recognition (Binford et al. 1987), medical diagnosis (Heckerman 1990), and search (Hansson & Mayer 1989). Heckerman (1995b) provides a review of recent applications of Bayesian Networks.

As Bayesian networks grew in popularity, their limitations became increasingly apparent. Although a powerful tool, BNs are not expressive enough for many real-world applications. More specifically, Bayesian Networks assume a simple attribute-value representation – that is, each problem instance involves reasoning about the same fixed number of attributes, with only the evidence values changing from problem instance to problem instance. This type of representation is inadequate for many problems of practical importance. Many domains require reasoning about varying numbers of related entities of different types, where the numbers, types and relationships among entities cannot be specified in advance and may themselves be uncertain. As will be demonstrated below, Bayesian networks are insufficiently expressive for such problems.

On the other hand, systems based on first-order logic (FOL) have the ability to represent entities of different types interacting with each other in varied ways. Sowa states that first-order logic "has enough expressive power to define all of mathematics, every digital computer that has ever been built, and the semantics of every version of logic, including itself" (Sowa 2000, page 41). For this reason, FOL has become the *de facto* standard for logical systems from both a theoretical and practical standpoint. However, systems based on classical first-order logic lack a theoretically principled, widely accepted, logically coherent methodology for reasoning under uncertainty.

As a result, a number of languages have appeared that extend the expressiveness of standard BNs in various ways (see section on related work below). As probabilistic languages become increasingly expressive, there is a need for a fuller characterization of their theoretical properties. Different communities appear to be converging around certain fundamental approaches to representing uncertain information about the attributes, behavior, and interrelationships

of structured entities (cf., Heckerman et al. 2004). This paper discusses some of the primary representational challenges that must be addressed by a logical formalism that combines first-order logic and probability. As a vehicle for presenting these ideas, we have chosen Multi-entity Bayesian networks (MEBN), a knowledge representation formalism that combines the expressive power of first-order logic with a sound and logically consistent treatment of uncertainty (Laskey 2005). MEBN syntax is designed to highlight the relationship between a MEBN theory and its first-order logic counterpart. Although our examples are presented using MEBN, our main focus is the underlying logical notions and not the language per se. That is, MEBN syntax should be viewed not as a competitor to other syntactic conventions such as plates or probabilistic relational models, but as a vehicle for expressing logical notions that cut across surface syntactic differences.

MEBN is not a computer language such as Java or C++, or an application such as Netica or Hugin (although it would be possible to construct software applications that implement MEBN). Rather, it is formal system that instantiates first-order Bayesian logic. That is, MEBN provides syntax, a set of model construction and inference processes, and semantics that together provide a means of defining probability distributions over unbounded and possibly infinite numbers of interrelated hypotheses. As such, MEBN provides a logical foundation for the many emerging languages that extend the expressiveness of Bayesian networks.

The purpose of this paper is to provide an accessible introduction to first-order probabilistic logic in general and MEBN in particular. In the context of a running example, we illustrate the limitations of standard BNs for situations that demand a more powerful representation formalism. We then gradually introduce additional elements into our example to illustrate the power of the additional representation capability provided by integrating first-order logic and probability.

Of Planets and Starships

We begin with a simple problem that can be modeled using standard BNs. Then, assuming the model as satisfactory for its purposes, we gradually expand it to embrace more general situations. Choosing a particular real-life domain would risk getting bogged down in domain-specific detail. For this reason, we opted to construct a case study based on the popular Paramount series *Star Trek*. Our examples have been constructed to be accessible to anyone having some familiarity with space-based science fiction.

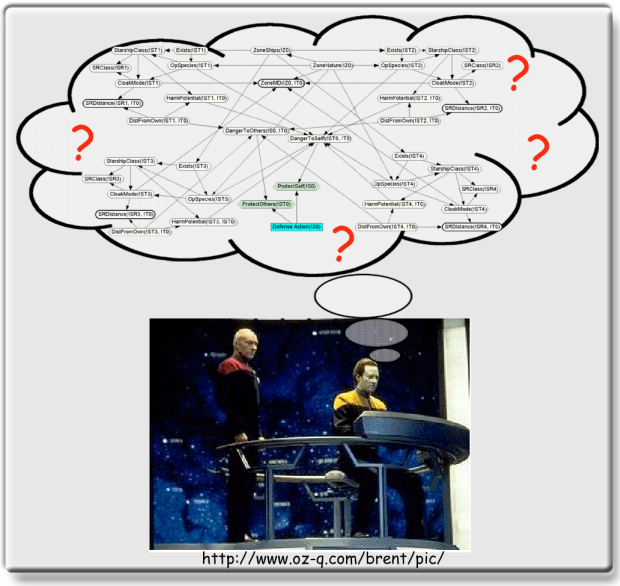

Figure 1 – Decision Support Systems in the 24th Century

A Simple BN Model

Figure 1 illustrates the operation of a 24th century decision support system tasked with helping Captain Picard to assimilate reports, assess their significance, and choose an optimal response. Of course, present-day systems are much less sophisticated than the system of Figure 1. We therefore begin our exposition narrating a highly simplified problem of detecting enemy starships.

In this simplified problem, the main task of a decision system is to model the problem of detecting Romulan starships (here considered as hostile by the United Federation of Planets) and assessing the level of danger they bring to our own starship, the *Enterprise*. All other starships were considered either friendly or neutral. Starship detection is performed by the *Enterprise's* suite of sensors, which can correctly detect and discriminate starships with an accuracy of 95%. However, Romulan starships could be in "cloak mode," which would make them invisible to the *Enterprise's* sensors. Even for the most current sensor technology, the only hint of a nearby starship in cloak mode is a slight magnetic disturbance caused by the enormous amount of energy required for cloaking. The *Enterprise* has a magnetic disturbance sensor, but

it is very hard to distinguish background magnetic disturbance from that generated by a nearby starship in cloak mode.

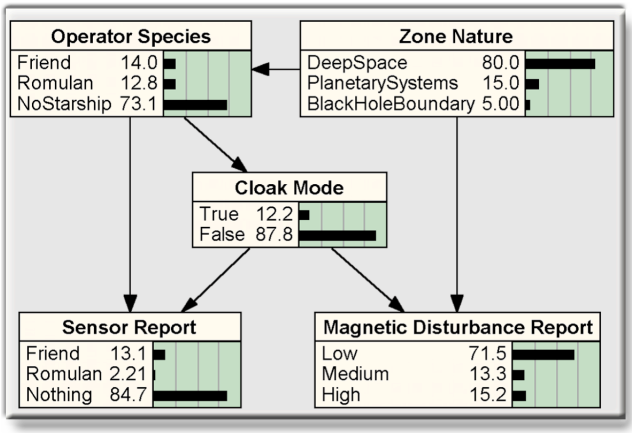

Figure 2 – The Basic Starship Bayesian Network

This simplified situation is modeled by the BN in Figure 2[1], which also considers the characteristics of the zone of space where the action takes place. Each node in our BN has a finite number of mutually exclusive, collectively exhaustive states. The node Zone Nature (ZN) is a root node, and its prior probability distribution can be read directly from Figure 2 (e.g. 80% for deep space). The probability distribution for Magnetic Disturbance Report (MDR) depends on the values of its parents ZN and Cloak Mode (CM). The strength of this influence is quantified via the conditional probability table (CPT) for node MDR, shown in Table 1. Similarly, Operator Species (OS) depends on ZN, and the two report nodes depend on CM and the hypothesis on which they are reporting.

Table 1 – Conditional Probability table for node MDR

Zone Nature	Cloak Mode	Magnetic Disturb. Rep.		
		Low	Medium	High
Deep Space	True	80.0	13.0	7.0
	False	85.0	10.0	5.0
Planetary Systems	True	20.0	32.0	48.0
	False	25.0	30.0	45.0
Black Hole Boundary	True	5.0	10.0	85.0
	False	6.9	10.6	82.5

Graphical models provide a powerful modeling framework and have been applied to many real world problems involving uncertainty. There is a large and growing literature on Bayesian network theory and applications (e.g. Charniak 1991, Jensen 1996, 2001, Neapolitan 1990, Oliver & Smith 1990, Pearl 1988).

[1] Bayesian network screen shots were constructed using Netica™, http://www.norsys.com.

How Complex Can We Go?

The model depicted above is of little use in a "real life" starship environment. After all, hostile starships cannot be expected to approach *Enterprise* one at a time so as to render its simple BN model usable. If four starships were closing in on the *Enterprise*, we would need to replace the BN of Figure 2 with the one shown in Figure 3. But even if we had a BN for each possible number of nearby starships, we still would not know which BN to use at any given time, because we don't know in advance how many starships the *Enterprise* is going to encounter. In short, BNs lack the expressive power to represent entity types (e.g., starships) that can be instantiated as many times as required for the situation at hand.

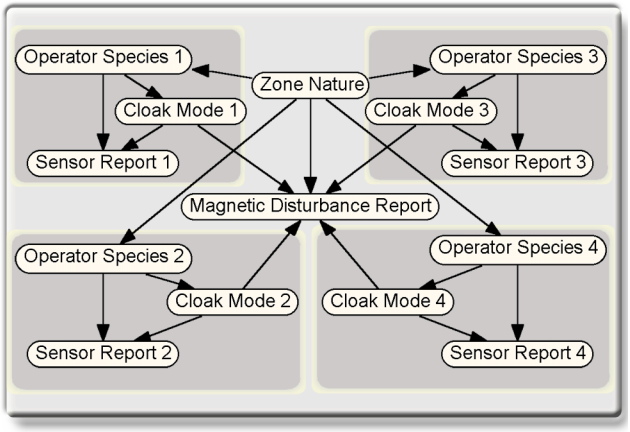

Figure 3 – The BN for Four Starships

In spite of its naiveté, let us briefly hold on to the premise that only one starship can be approaching the *Enterprise* at a time, so that the model of Figure 2 is valid. Furthermore, suppose we are traveling in deep space, our sensor report says there is no trace of a nearby starship (i.e. the state of node SR state is *Nothing*), and we receive a report of a strong magnetic disturbance (i.e. the state of node MDR is *High*). Table 1 shows that the likelihood ratio for a high MDR is 7/5 = 1.4 in favor of a starship in cloak mode. Although this favors a cloaked starship in the vicinity, the evidence is not overwhelming.

Repetition is a powerful way to boost the discriminatory power of weak signals. As an example from airport terminal radars, a single pulse reflected from an aircraft usually arrives back to the radar receiver very weakened, making it hard to set apart from background noise. However, a steady sequence

of reflected radar pulses is easily distinguishable from background noise. Following the same logic, it is reasonable to assume that an abnormal background disturbance will show random fluctuation, whereas a disturbance caused by a starship in cloak mode would show a characteristic temporal pattern. Thus, when there is a cloaked starship nearby, the MDR state at any time depends on its previous state. A BN similar to the one in Figure 4 could capitalize on this for pattern recognition purposes.

Dynamic Bayesian Networks (DBNs) allow nodes to be repeated over time (Murphy 1998). The model of Figure 4 has both static and dynamic nodes, and thus is a *partially* dynamic Bayesian network (PDBN), also known as a temporal Bayesian network (e.g. Takikawa et al. 2001). While DBNs and PDBNs are useful for temporal recursion, a more general recursion capability is needed, as well as a parsimonious syntax for expressing recursive relationships.

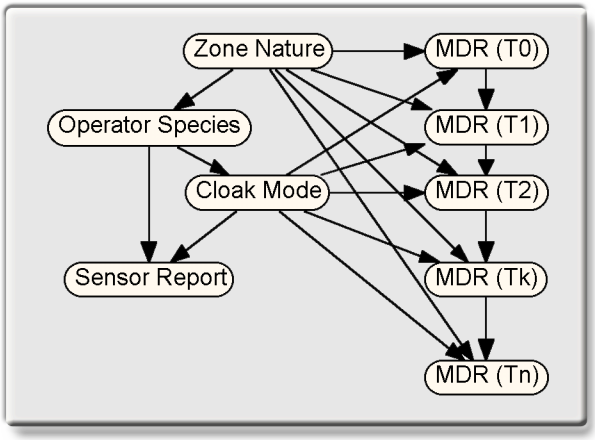

Figure 4 – The BN for One Starship with Recursion

This section has provided just a glimpse of the issues that confront an engineer attempting to apply Bayesian networks to realistically complex problems. The next section extends the complexity of our model to show how MEBN logic handles many of the difficulties commonly encountered in knowledge representation.

Using MEBN Logic

The limited model of the previous section would be of little use in increasing the Captain's awareness of the level of danger faced by the *Enterprise*. In addition to the model's naïve assumptions, there were clear omissions such as the assessment of the threat posed by a given starship, its ability and willingness to attack our own vessel, etc. These and other pertinent issues are addressed in the context of a richer scenario for which the power of MEBN is required.

A More "Realistic" Sci-fi Scenario

Like present-day Earth, 24th Century outer space is not a politically trivial environment. Our first extension introduces different alien species with diverse profiles. Although MEBN logic can represent the full range of species inhabiting the Universe in the 24th century, for purposes of this paper we prefer to use a simpler model. We therefore limit the explicitly modeled species to Friends[2], Cardassians, Romulans, and Klingons while addressing encounters with other possible races using the general label *Unknown*.

Cardassians are constantly at war with the Federation, so any encounter with them is considered a hostile event. Fortunately, they do not possess cloaking technology, which makes it easier to detect and discriminate them. Romulans, are more ambiguous, behaving in a hostile manner in roughly half their encounters with Federation starships. Klingons, which also possess cloaking technology, have a peace agreement with the Federation of Planets, but their treacherous and aggressive behavior makes them less reliable than friends. Finally, when facing an unknown species, the historical log of such events shows that out of every ten new encounters, only one was hostile.

Apart from the species of its operators, a truly "realistic" model would consider each starship's type, offensive power, the ability of inflict harm to the *Enterprise* given its range, and numerous other features pertinent to the model's purpose. We will address these issues as we present the basic constructs of MEBN logic.

Understanding MFrags

MEBN logic represents the world as comprised of entities that have attributes and are related to other entities. Random variables represent features of entities and relationships among entities. Knowledge about attributes and relationships is expressed as a collection of MEBN fragments (MFrags) organized into MEBN Theories (MTheories). An MFrag

[2] The interest reader can find further information on the Star Trek series in a plethora of websites dedicated to preserve or to extend the history of series, such as www.startrek.com, www.ex-astris-scientia.org, or techspecs.acalltoduty.com.

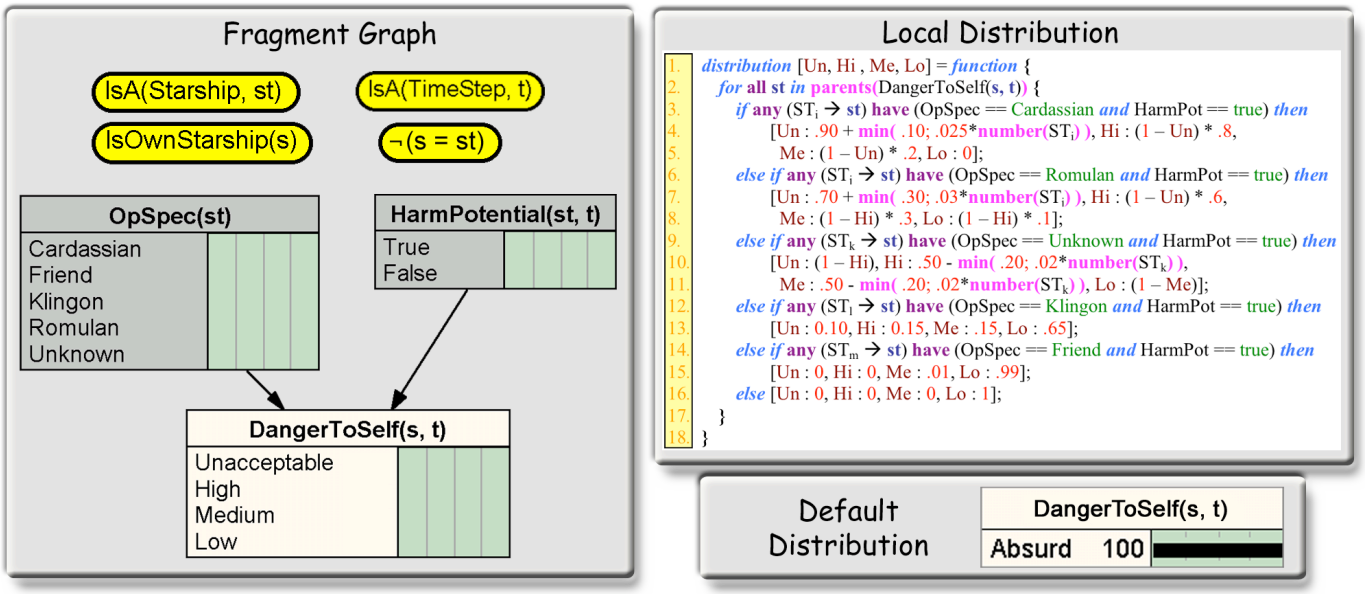

Figure 5 – The DangerToSelf MFrag

represents a conditional probability distribution for instances of its resident RVs given their parents in the fragment graph and the context nodes. An MTheory is a set of MFrags that collectively satisfies consistency constraints ensuring the existence of a unique joint probability distribution over instances of the RVs represented in each of the MFrags within the set.

Like a BN, an MFrag contains nodes, which represent RVs, arranged in a directed graph whose edges represent direct dependence relationships. An isolated MFrag can be roughly compared with a standard BN with known values for its root nodes and known local distributions for its non-root nodes. For example, the MFrag of Figure 5 represents knowledge about the degree of danger to which our own starship is exposed. The fragment graph has seven nodes. The four nodes at the top of the figure are context nodes; the two darker nodes below the context nodes are the input nodes; and the bottom node is a resident node.

A node in an MFrag may have a parenthesized list of arguments. These arguments are placeholders for entities in the domain. For example, the argument *st* to *HarmPotential*(*st, t*) is a placeholder for an entity that might harm us, while the argument *t* is a placeholder for the time step this instance represents. To refer to an actual entity in the domain, the argument is replaced with a *unique identifier*. By convention, unique identifiers begin with an exclamation point, and no two distinct entities can have the same unique identifier. By substituting unique identifiers for a RV's arguments, we can make *instances* of the RV. For example, *HarmPotential*(!*ST*1, !*T*1) and *HarmPotential*(!*ST*2, !*T*1) are two instances of *HarmPotential*(*st, t*) that both occur in the time step !*T*1.

The resident nodes of an MFrag have local distributions that define how their probabilities depend on the values of their parents in the fragment graph. In a complete MTheory, each random variable has exactly one *home MFrag*, where its local distribution is defined.[3] Input and context nodes (e.g., *OpSpec*(*st*) or *IsOwnStarship*(*s*)) influence the distribution of the resident nodes, but their distributions are defined in their own home MFrags.

Context nodes represent conditions that must be satisfied for the influences and local distributions of the fragment graph to apply. Context nodes are Boolean nodes: that is, they may have value *True*, *False*, or *Absurd*.[4] Context nodes having value *True* are said to be satisfied. As an example, if we substitute the unique identifier for the *Enterprise* (i.e., !*ST*0) for the variable *s* in *IsOwnStarship*(*s*), the resulting hypothesis will be true. If, instead, we

[3] Although standard MEBN logic does not support polymorphism, it could be extended to a typed polymorphic version that would permit a random variable to be resident in more than one MFrag.

[4] State names in this paper are alphanumeric strings beginning with a letter, including *True* and *False*. However, Laskey (2005) uses the symbols T for *True*, F for *False*, and ⊥ for *Absurd*, and requires other state names to begin with an exclamation point (because they are unique identifiers)

substitute a different starship unique identifier (say, !*ST*1), then this hypothesis will be false. Finally, if we substitute the unique identifier of a non-starship (say, !*Z*1), then this statement is absurd (i.e., it is absurd to ask whether or not a zone in space is one's own starship).

To avoid cluttering the fragment graph, we do not show the states of context nodes as we do with input and resident nodes, because they are Boolean nodes whose values are relevant only for deciding whether to use a resident random variable's local distribution or its default distribution.

No probability values are shown for the states of the nodes of the fragment graph in Figure 5. This is because nodes in a fragment graph do not represent individual random variables with well-defined probability distributions. Instead, a node in an MFrag represents a generic class of random variables. To draw inferences or declare evidence, we must create instances of the random variable classes.

To find the probability distribution for an instance of *DangerToSelf*(*s, t*), we first identify all instances of *HarmPotential*(*st, t*) and *OpSpec*(*st*) for which the context constraints are satisfied. If there are none, we use the default distribution that assigns value *Absurd* with probability 1. Otherwise, to complete the definition of the MFrag of Figure 5, we must specify a local distribution for its lone resident node, *DangerToSelf*(*s, t*). The pseudo-code of Figure 5 defines a local distribution for the danger to a starship due to all starships that influence its danger level. Local distributions in standard BNs are typically represented by static tables, which limits each node to a fixed number of parents. On the other hand, an instance of a node in an MTheory might have any number of parents. Thus, MEBN implementations (i.e. languages based on MEBN logic) must provide an expressive language for defining local distributions. We use pseudo-code to convey the idea of using local expressions to specify probability distributions, while not committing to a particular syntax.

Lines 3 to 5 cover the case in which there is at least one nearby starship operated by Cardassians and having the ability to harm the *Enterprise*. In this uncomfortable situation for our starship, the probability of an unacceptable danger to self is 0.90 plus the minimum of 0.10 and the result of multiplying 0.025 by the total number of starships that are harmful and operated by Cardassians. Also the remaining belief (i.e. the difference between 100% and the belief in state *Unacceptable* is divided between *High* (80% of the remainder) and *Medium* (20% of the remainder) whereas belief in *Low* is zero. The remaining lines use similar formulas to cover the other possible configurations in which there exist starships with potential to harm *Enterprise* (i.e. *HarmPotential*(*st, t*) = *True*).

The last conditional statement of the local expression covers the case in which no nearby starships can inflict harm upon the *Enterprise* (i.e. all nodes *HarmPotential*(*st, t*) have value *False*). In this case, the value for *DangerToSelf*(*s, t*) is *Low* with probability 1.

Figure 6 depicts an instantiation of the Danger To Self MFrag for which we have four starships nearby, three of them operated by Cardassians and one by the Romulans. Also, the Romulan and two of the Cardassian starships are within a range at which they can harm the *Enterprise*, whereas the other Cardassian starship is too far away to inflict any harm.

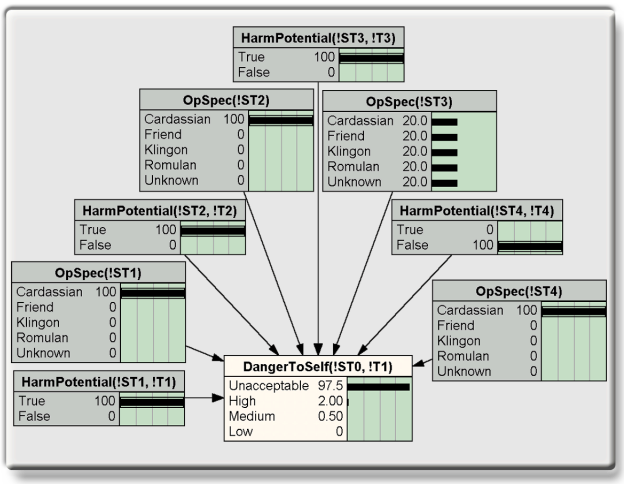

Figure 6 – An Instance of the DangerToSelf MFrag

Following the procedure described in Figure 5, the belief for state *Unacceptable* is .975 (.90 + .025*3) and the beliefs for states *High*, *Medium*, and *Low* are .02 ((1-.975)*.8), .005 ((1-.975)*.2), and zero respectively.

In short, the pseudo-code covers all possible input node configurations by linking the danger level to the number of nearby starships that have the potential to harm our own starship. The formulas state that if there are any Cardassians nearby, then the distribution for danger level given the number of Cardassians will be:

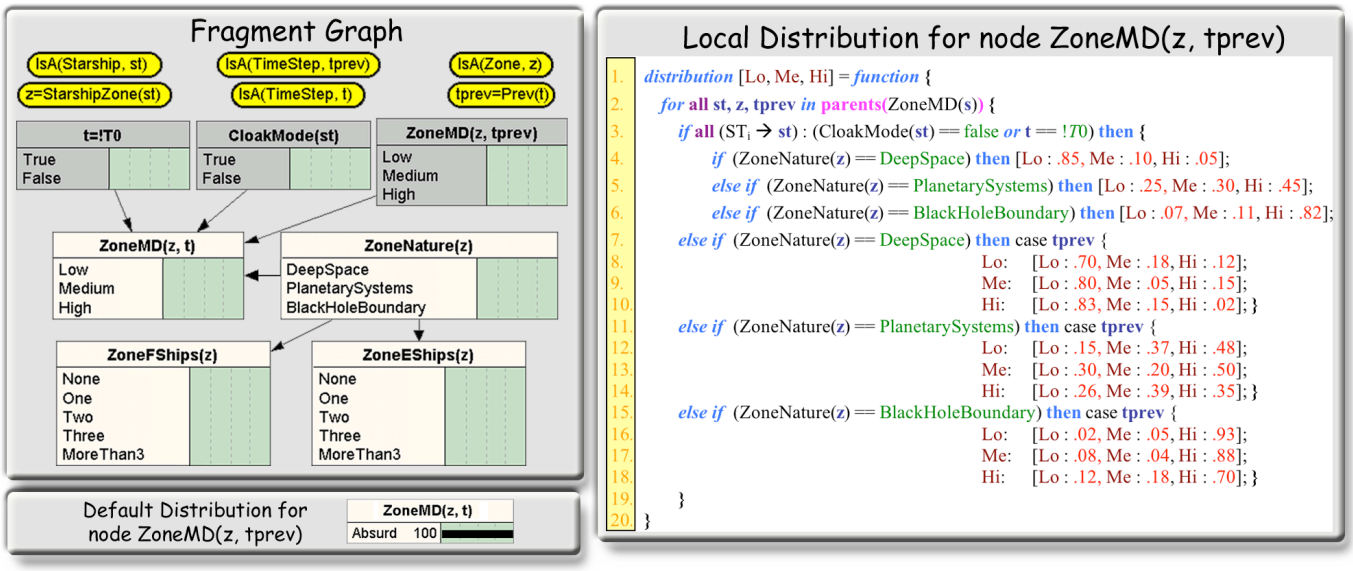

Figure 7 – The Zone MFrag

1 Cardassian ship - [0.925, 0.024, 0.006, 0];
2 Cardassian ships - [0.99, 0.008, 0.002, 0];
3 Cardassian ships - [0.975, 0.2, 0.05, 0];
4 or more Cardassian ships - [1, 0, 0, 0]

Also, if there are only Romulans with *HarmPot*(*s*) = *True*, then the distribution becomes:

1 Romulan ship - [.73, .162, .081, .027];
2 Romulan ships - [.76, .144, .072, .024];
... ,
10 or more Romulan ships - [1, 0, 0, 0]

For a situation in which only starships operated by unknown species can harm *Enterprise*, the probability distribution is more evenly distributed:

1 Unknown ship - [.02, .48, .48, .02];
2 Unknown ships - [.04, .46, .46, .04];
... ,
10 or more Unknown ships - [.20, .30, .30, .20]

Finally, if there are only friendly starships nearby with the ability to harm the *Enterprise*, then the distribution becomes [0, 0, 0.01, .99]. The last line indicates that if that no starship can harm the *Enterprise*, then the danger level will be *Low* for sure.

As noted previously, a powerful representational formalism is needed to represent complex scenarios at a reasonable level of fidelity. In our example, we could have added additional detail and explored many nuances. For example, a large number of nearby Romulan ships might indicate a coordinated attack and therefore indicate greater danger than an isolated Cardassian ship. Our example was purposely kept simple in order to clarify the basic capabilities of the logic. It is clear that more complex knowledge patterns could be accommodated as needed to suit the requirements of the application. MEBN logic has built-in logical MFrags that provide the ability to express anything that can be expressed in first-order logic. Laskey (2005) proves that MEBN logic can implicitly express a probability distribution over interpretations of any consistent, finitely axiomatizable first-order theory. This provides MEBN with sufficient expressive power to represent virtually any scientific hypothesis.

Recursive MFrags

One of the main limitations of BNs is their lack of support for recursion. Extensions such as dynamic Bayesian networks provide the ability to define certain kinds of recursive relationships. MEBN provides theoretically grounded support for very general recursive definitions of local distributions. Figure 7 depicts an example of how an MFrag can represent temporal recursion.

As we can see from the context nodes, in order for the local distribution to apply, *z* has to be a zone and *st* has to be a starship that has *z* as its current position. In addition, *tprev* and *t* must be *TimeStep* entities, and *tprev* is the step preceding *t*.

Other varieties of recursion can also be represented in MEBN logic by means of MFrags that allow influences between instances of the same random variable. Allowable recursive definitions must ensure that no random variable instance can influence its own probability distribution.

As in non-recursive MFrags, the input nodes in a recursive MFrag include nodes whose local

distributions are defined in another MFrag (i.e., *CloakMode*(*st*)). In addition, the input nodes may include instances of recursively-defined nodes in the MFrag itself. For example, the input node *ZoneMD*(*z*, *tprev*) represents the magnetic disturbance in zone *z* at the previous time step, which influences the current magnetic disturbance *ZoneMD*(*z*, *t*). The recursion is grounded by specifying an initial distribution at time !*T*0 that does not depend on a previous magnetic disturbance.

Figure 8 illustrates how recursive definitions can be applied to construct a *situation-specific Bayesian Network* (SSBN) to answer a query. Our query concerns the magnetic disturbance at time !*T*3 in zone !*Z*0, where !*Z*0 is known to contain our own uncloaked starship !*ST*0 and exactly one other starship !*ST*1, which is known to be cloaked. To build the graph shown in this picture, we begin by creating an instance of the home MFrag of the query node *ZoneMD*(!*Z*0,!*T*3). That is, we substitute !*Z*0 for *z* and !*T*3 for *t*, and then create all instances of the remaining random variables that meet the context constraints. Next, we build any CPTs we can already build. CPTs for *ZoneMD*(!*Z*0,!*T*3), *ZoneNature*(!*Z*0), *ZoneEShips*(!*Z*0), and *ZoneFShips*(!*Z*0) can be constructed because they are resident in the retrieved MFrag. Single-valued CPTs for *CloakMode*(!*ST*0), *CloakMode*(!*ST1*), and !*T*3=!*T*0 can be specified because the values of these random variables are known.

This leaves us with one node, *ZoneMD*(!*Z*0,!*T*2), for which we have no CPT. To construct its CPT, we must retrieve its home MFrag, and instantiate any random variables that meet its context constraints and have not already been instantiated. The new random variables created in this step are *ZoneMD*(!*Z*0,!*T*1) and !*T*2=!*T*0. We know the value of the latter, and we retrieve the home MFrag of the former. This process continues until we have added all the nodes of Figure 8. At this point we can construct CPTs for all random variables, and the SSBN is complete.[5]

The MFrag depicted in Figure 7 defines the local distribution that applies to all these instances, even though for brevity we only displayed the probability distributions (local and default) for node *ZoneMD*(z, *t*). Note that when there is no starship with cloak mode activated, the probability distribution for magnetic disturbance given the zone nature does not change with time. When there is at least one starship with cloak mode activated, then the magnetic disturbance tends to fluctuate regularly with time in the manner described by the local expression. For the sake of simplicity, we assumed that the local distribution depends only on whether there is a cloaked starship nearby.

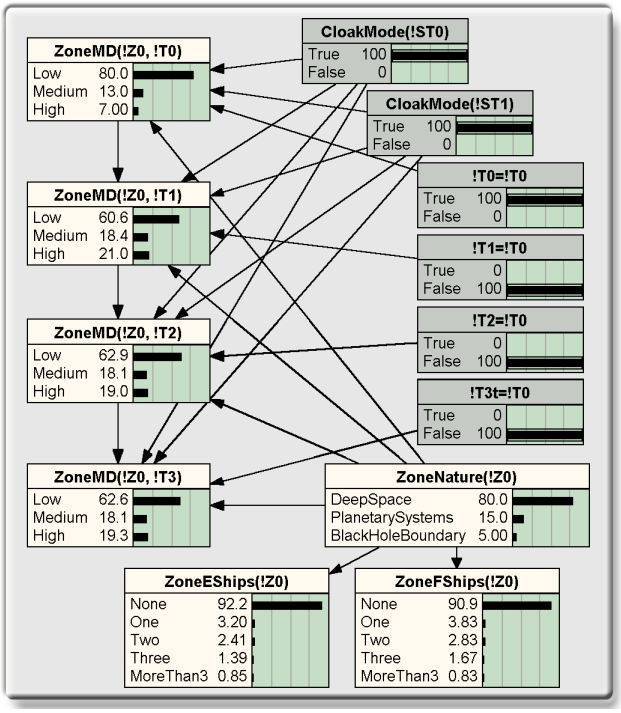

Figure 8 – SSBN Constructed from Zone MFrag

We also assumed that the initial distribution for the magnetic disturbance when there are cloaked starships is equal to the stationary distribution given the zone nature and the number of cloaked starships present initially. Of course, it would be possible to write different local expressions expressing a dependence on the number of starships, their size, their distance from the *Enterprise*, etc.

MFrags provide a flexible means to represent knowledge about specific subjects within the domain of discourse, but the true gain in expressive power is revealed when we aggregate these "knowledge patterns" to form a coherent model of the domain of discourse that can be instantiated to reason about

[5] For efficiency reasons, most knowledge-based model construction systems would not explicitly represent root evidence nodes such as *CloakMode*(!*ST*0) or !*T*1=!*T*0 or barren nodes such as *ZoneFShips*(!*Z*0) and *ZoneFShips*(!*Z*0). For expository purposes, we have taken the logically equivalent, although less computationally efficient, approach of including all these nodes explicitly.

[7] The alert reader may notice that root evidence nodes and barren nodes that were included in the constructed network of Figure 8 are not included here. As noted above, explicitly representing these nodes is not necessary.

specific situations and refined through learning. It is important to note that just collecting a set MFrags that represent specific parts of a domain is not enough to ensure a coherent representation of that domain. For example, it would be easy to specify a set of MFrags with cyclic influences, or one having multiple conflicting distributions for a random variable in different MFrags. The following section describes how to define complete and coherent domain models as collections of MFrags.

Building MEBN models with MTheories

In order to build a coherent model we have to make sure that our set of MFrags collectively satisfies consistency constraints ensuring the existence of a unique joint probability distribution over instances of the random variables mentioned in the MFrags. Such a coherent collection of MFrags is called an MTheory. An MTheory represents a joint probability distribution for an unbounded, possibly infinite number of instances of its random variables. This joint distribution is specified by the local and default distributions within each MFrag together with the conditional independence relationships implied by the fragment graphs.

The MFrags described above are part of a *generative MTheory* for the intergalactic conflict domain. A generative MTheory summarizes statistical regularities that characterize a domain. These regularities are captured and encoded in a knowledge base using some combination of expert judgment and learning from observation. To apply a generative MTheory to reason about particular scenarios, we need to provide the system with specific information about the individual entity instances involved in the scenario. On receipt of this information, we can use Bayesian inference both to answer specific questions of interest (e.g., how high is the current level of danger to the *Enterprise*?) and to refine the MTheory (e.g., each encounter with a new species gives us additional statistical data about the level of danger to the *Enterprise* from a starship operated by an unknown species). Bayesian inference is used to perform both problem-specific inference and learning in a sound, logically coherent manner.

Findings are the basic mechanism for incorporating observations into MTheories. A finding is represented as a special 2-node MFrag containing a node from the generative MTheory and a node declaring one of its states to have a given value. From a logical point of view, inserting a finding into an MTheory corresponds to asserting a new axiom in a first-order theory. In other words, MEBN logic is inherently open, having the ability to incorporate new axioms as evidence and update the probabilities of all random variables in a logically consistent way.

In addition to the requirement that each random variable must have a unique home MFrag, a valid MTheory must ensure that all recursive definitions terminate in finitely many steps and contain no circular influences. Finally, as we saw above, random variable instances may have a large, and possibly unbounded number of parents. A valid MTheory must satisfy an additional condition to ensure that the local distributions have reasonable limiting behavior as more and more parents are added. Laskey (2005) proved that when an MTheory satisfies these conditions (as well as other technical conditions that are unimportant to our example), then there exists a joint probability distribution on the set of instances of its random variables that is consistent with the local distributions assigned within its MFrags. Furthermore, any consistent, finitely axiomatizable FOL theory can be translated to infinitely many MTheories, all having the same purely logical consequences, that assign different probabilities to statements whose truth-value is not determined by the axioms of the FOL theory. MEBN logic contains a set of built-in logical MFrags (including quantifier, indirect reference, and Boolean connective MFrags) that provide the ability to represent any sentence in first-order logic. If the MTheory satisfies additional conditions, then a conditional distribution exists given any finite sequence of findings that does not logically contradict the logical constraints of the generative MTheory. MEBN logic thus provides a logical foundation for systems that reason in an open world and incorporate observed evidence in a mathematically sound, logically coherent manner.

Figure 9 shows an example of a generative MTheory for our *Star Trek* domain. For the sake of conciseness, the local distribution formulas and the default distributions are not shown here.

The Entity Type, at the right side of Figure 9, is meant to formally declare the possible types of entity that can be found in the model. This is a generic MFrag that allows the creation of domain-oriented types (which are represented by TypeLabel entities) and forms the basis for a Typed system. In our simple model we did not address the creation or the explicit support for entity types. Standard MEBN logic as defined in Laskey (2005) is untyped, meaning that a

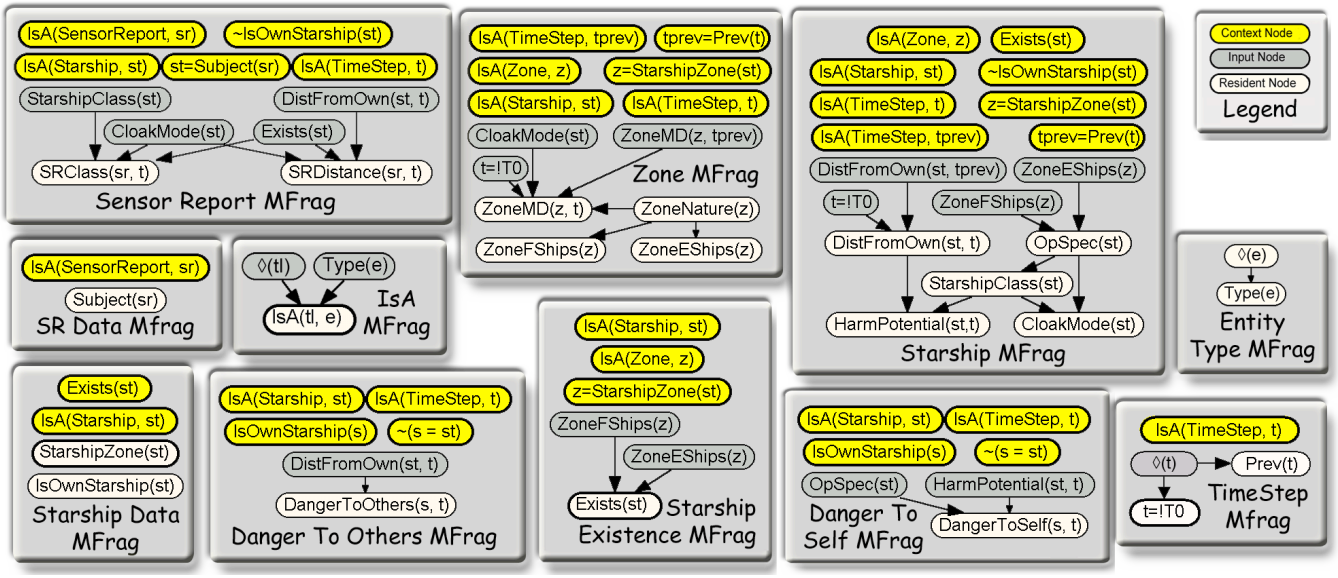

Figure 9 – The Star Trek Generative MTheory

knowledge engineer who wishes to represent types must explicitly define the necessary logical machinery. The Entity Type MFrag of Figure 9 defines an extremely simple kind of type structure. MEBN can be extended with MFrags to accommodate any flavor of typed system, including more complex capabilities such as sub-typing, polymorphism, multiple-inheritance, etc.

It is important to understand the power and flexibility that MEBN logic gives to knowledge base designers by allowing multiple, equivalent ways of portraying the same knowledge. Indeed, the generative MTheory of Figure 9 is just one of the many possible (consistent) sets of MFrags that can be used to represent a given joint distribution. There, we attempted to cluster the random variables in a way that naturally reflects the structure of the objects in that scenario (i.e. we adopted an object oriented approach to modeling), but this was only one design option among the many allowed by the logic. As an example of such flexibility, Figure 10 depicts the same knowledge contained in the Starship MFrag of Figure 9 (right side) using three different MFrags. In this case, the modeler might have opted for decomposing an MFrag in order to get the extra flexibility of smaller, more specific MFrags that can be combined in different ways. Another knowledge engineer might prefer the more concise approach of having all knowledge in just one MFrag. Ultimately, the approach to be taken when building an MTheory will depend on many factors, including the model's purpose, the background and preferences of the model's stakeholders, the need to interface with external systems, etc.

First Order Logic (or one of its subsets) provides the theoretical foundation for the type systems used in popular object-oriented and relational languages. MEBN logic provides the basis for extending the capability of these systems by introducing a sound mathematical basis for representing and reasoning under uncertainty. Among the advantages of a MEBN-based typed system is the ability to represent *type uncertainty*. As an example, suppose we had two different types of space traveling entities, starships and comets, and we are not sure about the type of a given entity. In this case, the result of a query that depends on the entity type will be a weighted average of the result given that the entity is a comet and the result given that it is a starship. Further advantages of a MEBN-based type system include the ability to refine type-specific probability distributions using Bayesian learning, assign probabilities to possible values of unknown attributes, reason coherently at multiple levels of resolution, and other features related to representing and reasoning with incomplete and/or uncertain information.

Another powerful aspect of MEBN, the ability to support *finite or countably infinite recursion*, is illustrated in the Sensor Report and Zone MFrags, both of which involve temporal recursion. The Time Step MFrag includes a formal specification of the local distribution for the initial step of the time recursion (i.e. when t=!T0) and of its recursive steps (i.e. when t does not refer to the initial step). Other

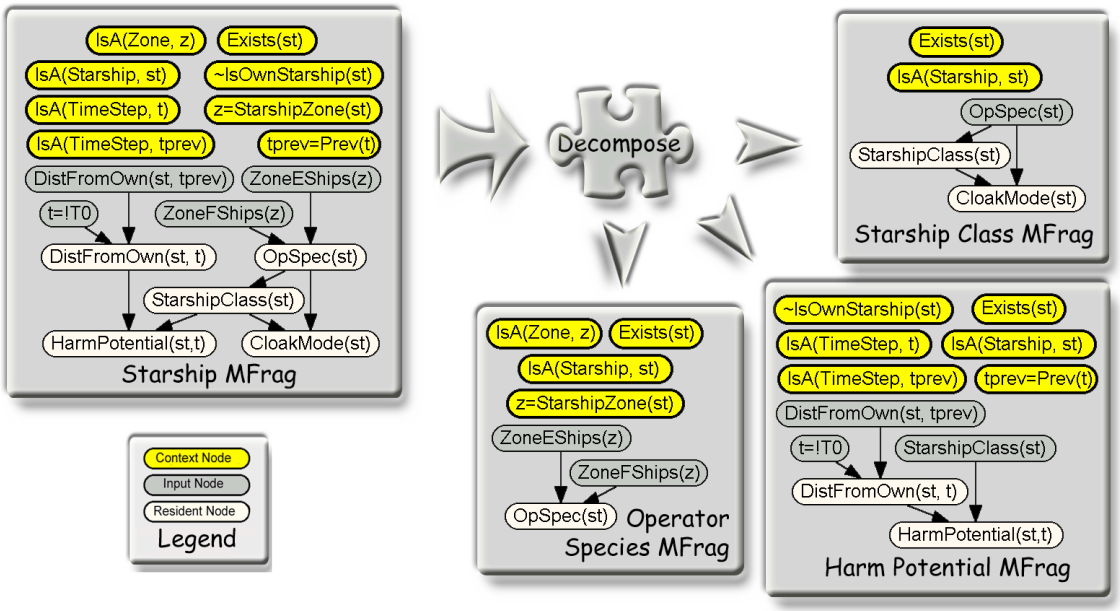

Figure 10 – Equivalent MFrag Representations of Knowledge

kinds of recursion can be represented in a similar manner.

MEBN logic also has the ability to represent and reason about hypothetical entities. Uncertainty about whether a hypothesized entity actually exists is called *existence uncertainty*. In our example model, the random variable *Exists*(*st*) is used to reason about whether its argument is an actual starship. For example, we might be unsure whether a sensor report corresponds to one of the starships we already know about, a starship of which we were previously unaware, or a spurious sensor report.

In this case, we can create a starship instance, say !*S*4, and assign a probability of less than 1.0 that *Exists*(!*S*4) has value *True*. Then, any queries involving !*S*4 will return results weighted appropriately by our belief in the existence of !*S*4. Furthermore, our belief in *Exists*(!*S*4) is updated by Bayesian conditioning as we obtain more evidence relevant to whether !*S*4 denotes a previously unknown starship. Representing existence uncertainty is particularly useful for counterfactual reasoning and reasoning about causality (Druzdzel & Simon 1993, Pearl 2000).

Because the *Star Trek* model was designed to demonstrate the capabilities of MEBN logic, we avoided issues that can be handled by the logic but would make the model too complex. As an example, one aspect that our model does not consider is *association uncertainty*, a very common problem in multi-sensor data fusion systems. Association uncertainty means that we are not sure about the source of a given report (e.g. whether a given report refers to starship !*S*4, !*S*2 or !*S*1). Many weakly discriminatory reports coming from possibly many starships produces an exponential set of combinations that require special *hypothesis management* methods (c.f. Stone et al. 1999). In the *Star Trek* model we avoided these problems by assuming our sensor suite can achieve perfect discrimination. However, the logic can represent and reason with association uncertainty, and thus provides a sound logical foundation for hypothesis management in multi-source fusion.

Making Decisions with MEBN Logic

Captain Picard has more than an academic interest in the danger from nearby starships. He must make decisions with life and death consequences. Multi-Entity Decision Graphs (MEDGs, or "medges") extend MEBN logic to support decision making under uncertainty. MEDGs are related to MEBNs in the same way influence diagrams are related to Bayesian Networks. A MEDG can be applied to any problem that involves optimal choice from a set of alternatives subject to given constraints.

When a decision MFrag (i.e. one that has decision and utility nodes) is added to a generative MTheory such as the one portrayed in Figure 9, the result is a MEDG. As an example, Figure 11 depicts a decision MFrag representing Captain Picard's choice of which defensive action to take. The decision node

DefenseAction(*s*) represents the set of defensive actions available to the Captain (in this case, to fire the ship's weapons, to retreat, or to do nothing). The value nodes capture Picard's objectives, which in this case are to protect *Enterprise* while also avoiding harm to innocent people as a consequence of his defensive actions. Both objectives depend upon Picard's decision, while *ProtectSelf*(*s*) is influenced by the perceived danger to *Enterprise* and *ProtectOthers*(*s*) is depends on the level of danger to other starships in the vicinity.

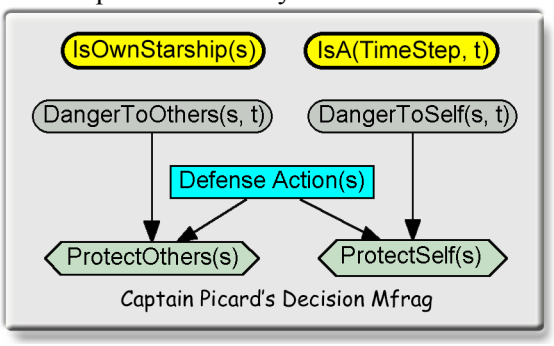

Figure 11 – The Star Trek Decision MFrag

The model described here is clearly an oversimplification of any "real" scenario a Captain would face. Its purpose is to convey the core idea of extending MEBN logic to support decision-making. Indeed, a more common situation is to have multiple, mutually influencing, often conflicting factors that together form a very complex decision problem, and require trading off different attributes of value. For example, a decision to attack would mean that little power would be left for the defense shields; a retreat would require aborting a very important mission.

MEDGs provide the necessary foundation to address all the above issues. Readers familiar with influence diagrams will appreciate that the main concepts required for a first-order extension of decision theory are all present in Figure 11. In other words, MEDGs have the same core functionality and characteristics of common MFrags. Thus, the utility table in *Survivability*(*s*) refers to the entity whose unique identifier substitutes for the variable *s*, which according to the context nodes should be our own starship (*Enterprise* in this case). Likewise, the states of input node *DangerToSelf*(*s, t*) and the decision options listed in *DefenseAction*(*s*) should also refer to the same entity.

Of course, this confers to MEDGs the expressive power of MEBN models, which includes the ability to use this same decision MFrag to model the decision process of the Captain of another starship. Notice that a MEDG Theory should also comply with the same consistency rules of standard MTheories, along with additional rules required for influence diagrams (e.g., value nodes are deterministic and must be leaf nodes or have only value nodes as children).

In our example, adding the Star Trek Decision MFrag of Figure 11 to the generative MTheory of Figure 9 will maintain the consistency of the latter, and therefore the result will be a valid generative MEDG Theory. Our simple example can be extended to more elaborate decision constructions, providing the flexibility to model decision problems in many different applications spanning diverse domains.

Inference in MEBN Logic

A generative MTheory provides prior knowledge that can be updated upon receipt of evidence represented as finding MFrags. We now describe the process used to obtain posterior knowledge from a generative MTheory and a set of findings.

In a BN model such as the ones shown in Figures 2 through 4, assessing the impact of new evidence involves conditioning on the values of evidence nodes and applying a belief propagation algorithm. When the algorithm terminates, beliefs of all nodes, including the node(s) of interest, reflect the impact of all evidence entered thus far. This process of entering evidence, propagating beliefs, and inspecting the posterior beliefs of one or more nodes of interest is called a query.

MEBN inference works in a similar way (after all, MEBN is a Bayesian logic), but following a more complex yet more flexible process. Whereas BNs are static models that must be changed whenever the situation changes (e.g. number of starships, time recursion, etc.), an MTheory implicitly represents an infinity of possible scenarios. In other words, the MTheory represented in Figure 9 (as well as the MEDG obtained by aggregating the MFrag in Figure 11) is a model that can be used for as many starships as we want, and for as many time steps we are interested in, for as many situations as we face from the 24th Century into the future.

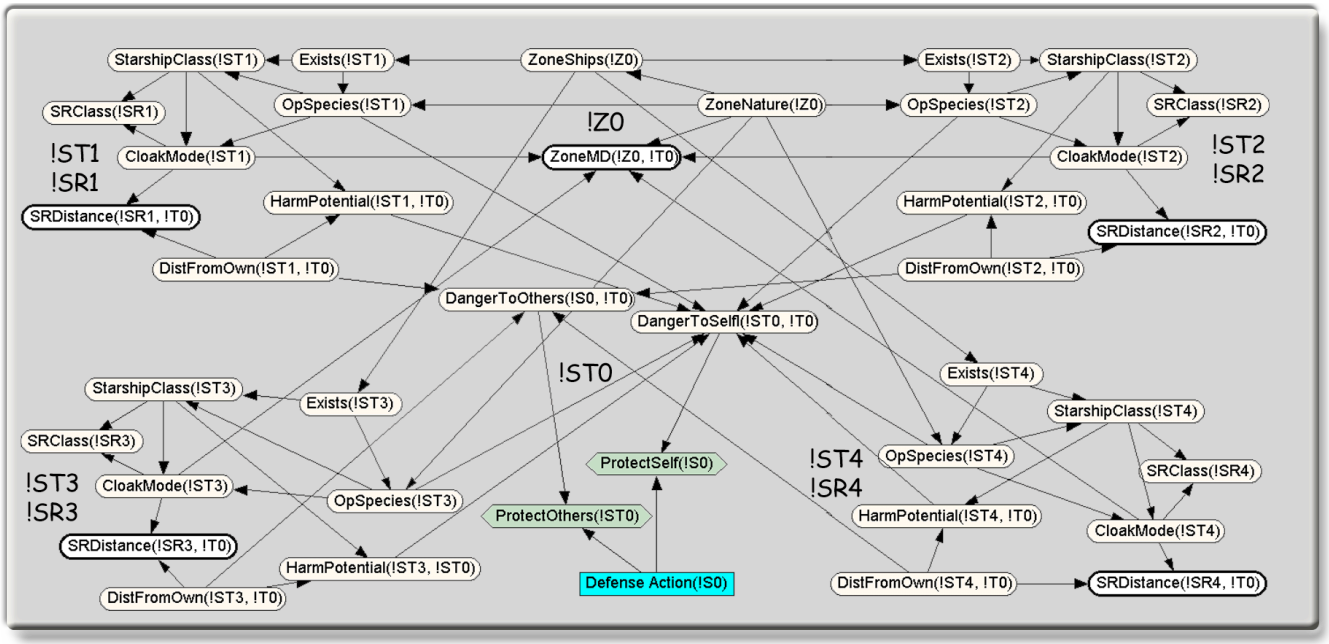

Figure 12 – SSBN for the Star Trek MTheory with Four Starships within Enterprise's Range

That said, the obvious question is how to perform queries within such a model. A simple example of query processing was given above in the section on temporal recursion. Here, we describe the general algorithm for constructing a situation-specific Bayesian network (SSBN). To do so, we have to have an initial generative MTheory (or MEDG Theory), a Finding set (which conveys particular information about the situation) and a Target set (which indicates the nodes of interest to us). For comparison, let's suppose we have a situation that is similar to the one in Figure 3, where four starships are within the *Enterprise*'s range. In that particular case, a BN was used to represent the situation at hand, which means we have a model that is "hardwired" to a known number (four) of starships, and any other number would require a different model. A standard Bayesian inference algorithm applied to that model would involve entering the available information about these four starships (i.e., the four sensor reports), propagating the beliefs, and obtaining posterior probabilities for the hypotheses of interest (e.g., the four *Starship Type* nodes).

Similarly, MEBN inference begins when a query is posed to assess the degree of belief in a target random variable given a set of evidence random variables. We start with a generative MTheory, add a set of finding MFrags representing problem-specific information, and specify the target nodes for our query. The first step in MEBN inference is to construct the SSBN, which can be seen as an ordinary Bayesian network constructed by creating and combining instances of the MFrags in the generative MTheory. Next, a standard Bayesian network inference algorithm is applied. Finally, the answer to the query is obtained by inspecting the posterior probabilities of the target nodes. A MEBN inference algorithm is provided in Laskey (2005). The algorithm presented there does not handle decision graphs, and so we will extend it slightly for purposes of illustrating how our MEDG Theory can be used to support the Captain's decision.

In our example, the finding MFrags will convey information that we have five starships (!*ST*0 through !*ST*4) and that the first is our own starship. For the sake of illustration, let's assume that our Finding set also includes data regarding the nature of the space zone we are in (!*Z*0), its magnetic disturbance for the first time step (!*T*0), and sensor reports for starships !*SR*1 to !*SR*4 for the first two time steps.

We assume that the Target set for our illustrative query includes an assessment of the level of danger experienced by the *Enterprise* and the best decision to take given this level of danger.

Figure 12 shows a situation-specific Bayesian network for our query[7]. To construct the SSBN, we begin by creating instances of the random variables in the Target set and the random variables for which we have findings. The target random variables are *DangerLevel*(!*ST*0) and *DefenseAction*(!*ST*0). The

finding random variables are the eight *SRDistance* nodes (2 time steps for each of four starships) and the two *ZoneMD* reports (one for each time step). Although each finding MFrag contains two nodes, the random variable on which we have a finding and a node indicating the value to which it is set, we include only the first of these in our situation-specific Bayesian network, and declare as evidence that its value is equal to the observed value indicated in the finding MFrag.

The next step is to retrieve and instantiate the home MFrags of the finding and target random variables. When each MFrag is instantiated, instances of its random variables are created to represent known background information, observed evidence, and queries of interest to the decision maker. If there are any random variables with undefined distributions, then the algorithm proceeds by instantiating their respective home MFrags. The process of retrieving and instantiating MFrags continues until there are no remaining random variables having either undefined distributions or unknown values. The result, if this process terminates, is the *SSBN* or, in our case, a *situation-specific decision graph* (SSDG). In some cases the SSBN can be infinite, but under conditions given in Laskey (2005), the algorithm produces a sequence of approximate SSBNs for which the posterior distribution of the target nodes converges to their posterior distribution given the findings. Mahoney and Laskey (1998) define a SSBN as a minimal Bayesian network sufficient to compute the response to a query. A SSBN may contain any number of instances of each MFrag, depending on the number of entities and their interrelationships. The SSDG in Figure 12 is the result of applying this process to the MEDG Theory in Figures 9 and 11 with the Finding and Target set we just defined.

Another important use for the SSBN algorithm is to help in the task of performing Bayesian learning, which is treated in MEBN logic as a sequence of MTheories.

Learning from Data

Learning graphical models from observations is usually divided in two different categories: inferring the parameters of the local distributions when the structure is known, and inferring the structure itself. In MEBN, by structure we mean the possible values of the random variables, their organization into MFrags, the fragment graphs, and the functional forms of the local distributions.

Figure 13 shows an example of parameter learning in MEBN logic in which we adopt the assumption that one can infer the length of a starship on the basis of the average length of all starships. This generic domain knowledge is captured by the generative MFrag, which specifies a prior distribution based on what we know about starship lengths.

One strong point about using Bayesian models in general and MEBN logic in particular is the ability to refine prior knowledge as new information becomes available. In our example, let's suppose that we receive precise information on the length of starships !*ST*2, !*ST*3, and !*ST*5; but have no information regarding the incoming starship !*ST*8.

The first step of this simple parameter learning example is to enter the available information to the model in the form of findings (see box StarshipLenghInd Findings). Then, we pose a query on the length of !*ST*8. The SSBN algorithm will instantiate all the random variables that are related to the query at hand until it finishes with the SSBN depicted in Figure 13 (box SSBN with Findings). In this example, the MFrags satisfy graph-theoretic conditions under which a re-structuring operation called *finding absorption* (Buntine 1994b) can be applied without changing the structure of the MFrags. Therefore, the prior distribution of the random variable *GlobalAvgLength* can be replaced by the posterior distribution obtained when adding evidence in the form of findings.

As a result of this learning process, the probability distribution for *GlobalAvgLength* has been refined in light of the new information conveyed by the findings. The resulting, more precise distribution can now be used not only to predict the length of !*ST*8 but for future queries as well. In our specific example, the same query would retrieve the SSBN in the lower right corner of Figure 13 (box SSBN with Findings Absorbed). One of the major advantages of the finding absorption operation is that it greatly improves the tractability of both learning and SSBN inference. We can also apply finding absorption to modify the generative MFrags themselves, thus creating a new generative MTheory that has the same conditional distribution given its findings as our original MTheory. In this new MTheory, the distribution of *GlobalAvgLength* has been modified to incorporate the observations and the finding random variables are set with probability 1 to their observed values. Restructuring MTheories via finding absorption can increase the efficiency of SSBN construction and of inference.

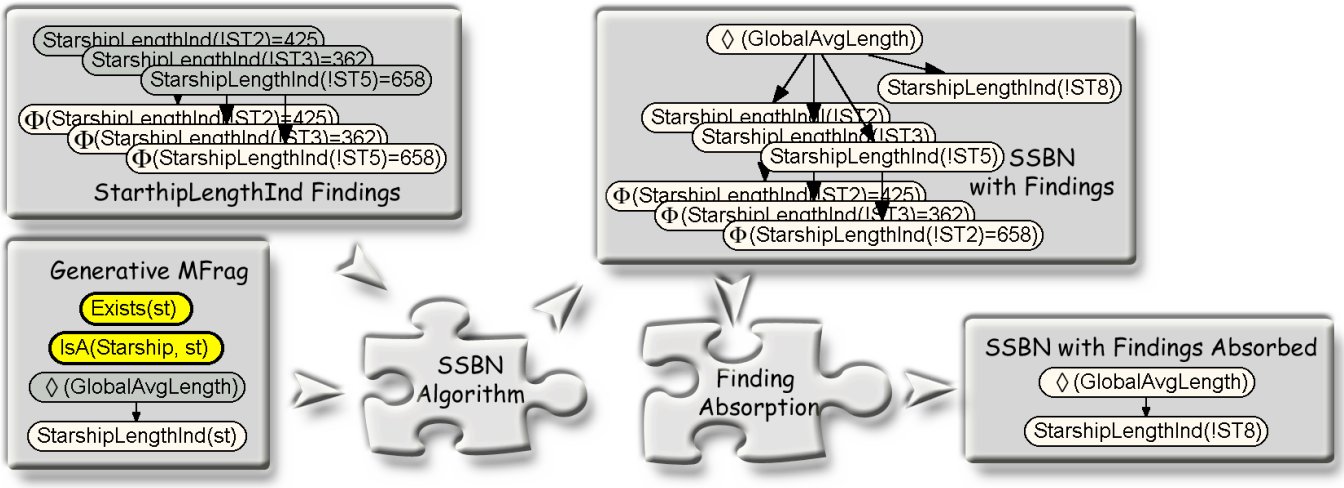

Figure 13 – Parameter Learning in MEBN

Structure learning in MEBN works in a similar fashion. As an example, let's suppose that when analyzing the data that was acquired in the parameter learning process above, a domain expert raises the hypothesis that the length of a given starship might depend on its class. To put it into a "real-life" perspective, let's consider two classes: Explorers and Warbirds. The first usually are vessels crafted for long distance journeys with a relatively small crew and payload. Warbirds, on the other hand, are heavily armed vessels designed to be flagships of a combatant fleet, usually carrying lots of ammunition, equipped with many advanced technology systems and a large crew. Therefore, our expert thinks it likely that the average length of Warbirds may be greater than the average length of Explorers.

In short, the general idea of this simple example is to mimic the more general situation in which we have a potential link between two attributes (i.e. starship length and class) but at best weak evidence to support the hypothesized correlation. This is a typical situation in which Bayesian models can use incoming data to learn both structure and parameters of a domain model. Generally speaking, the solution for this class of situations is to build two different structures and apply Bayesian inference to evaluate which structure is more consistent with the data as it becomes available.

The initial setup of the structure learning process for this specific problem is depicted in Figure 14. Each of our two possible structures is represented by its own generative MFrag. The first MFrag is the same as before: the length of a starship depended only on a global average length that applied to starships of all classes. The upper left MFrag of Figure 14, StarshipLengthInd MFrag conveys this hypothesis. The second possible structure, represented by the ClassAvgLength and Starship-LengthDep MFrags, covers the case in which a starship class influences its length.

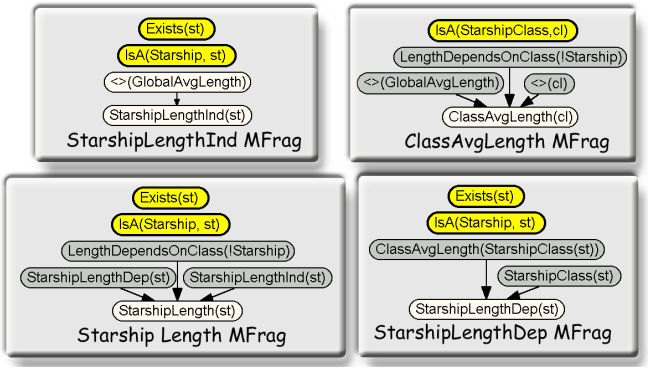

Figure 14 – Structure Learning in MEBN

The two structures are then connected by the Starship Length MFrag, which has the format of a *multiplexor* MFrag. The distribution of a multiplexor node such as *StarshipLength*(*st*) always has one parent *selector* node defining which of the other parents is influencing the distribution at a given situation.

In this example, where we have only two possible structures, the selector parent will be a two-state node. Here, the selector parent is the Boolean *LengthDependsOnClass*(!*Starship*). When this node has value *False* then *StarshipLength*(*cl*) will be equal to *StarshipLengthInd*(*st*), the distribution of which does not depend on the starship's class. Conversely, if the selector parent has value *True* then *StarshipLength*(*cl*) will be equal to *StarshipLength-*

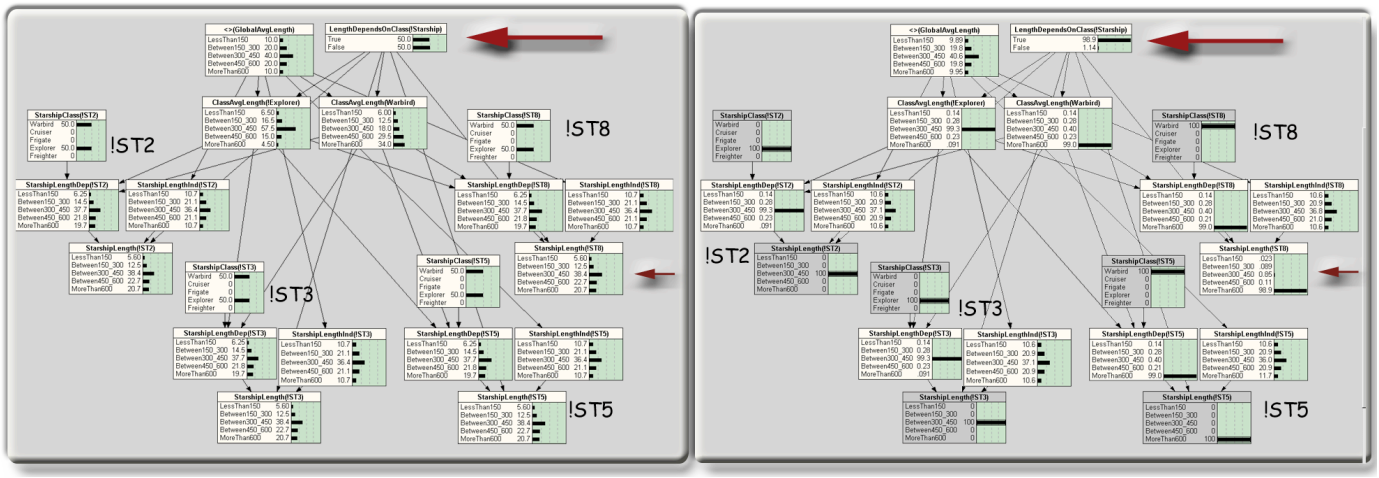

Figure 15 – SSBNs for the Parameter Learning Example

Dep(*st*), which is directly influenced by *ClassAvg-Length*(*StarshipClass*(*st*)).

Figure 15 shows the result of applying the SSBN algorithm to the generative MFrags in Figure 14. The SSBN on the left doesn't have the findings included, but only information about the existence of four starships. It can be noted that we choose our prior for the selector parent (the Boolean node on the top of the SSBN) to be the uniform distribution, which means we assumed that both structures (i.e. class affecting length or not) have the same prior probability.

For the SSBN in the right side we included the known facts that !*ST*2 and !*ST*3 belong to the class of starships !*Explorer*, and that !*ST*5 and !*ST*8 are Warbird vessels. Further, we included the lengths of three ships for which we have length reports. The result of the inference process was not only an estimate of the length of !*ST*8 but a clear confirmation that the data available strongly supports the hypothesis that the class of a starship directly influences its length.

It may seem cumbersome to define different random variables, *StarshipLengthInd* and *Starship-LengthDep*, for each hypothesis about the influences on a starship's length. As the number of structural hypotheses becomes large, this can become quite unwieldy. Fortunately, we can circumvent this difficulty by introducing a typed version of MEBN and allowing the distributions of random variables to depend on the type of their argument. A detailed presentation of typed MEBN is beyond the scope of this paper.

This basic construction is compatible with the standard approaches to Bayesian structure learning in graphical models (e.g. Cooper & Herskovits 1992, Friedman & Koller 2000, Heckerman *et al.* 1995a, Jordan 1999).

Unifying Classical Logic and Probability

In classical logic, the most that can be said about a hypothesis that can be neither proven nor disproven is that its truth-value is unknown. Practical reasoning demands more. Captain Picard's life depends on assessing the plausibility of many hypotheses he can neither prove nor disprove. Yet, he also needs first-order logic's ability to express generalizations about properties of and relationships among entities. In short, he needs a probabilistic logic with first-order expressive power.

Although there have been many attempts to integrate classical first-order logic with probability, MEBN is the first fully first-order Bayesian logic (Laskey, 2005). MEBN logic can assign probabilities in a logically coherent manner to any set of sentences in first-order logic, and can assign a conditional probability distribution given any consistent set of finitely many first-order sentences. That is, anything that can expressed in first-order logic can be assigned a probability by MEBN logic. The probability distribution represented by an MTheory can be updated via Bayesian conditioning to incorporate any finite sequence of findings that are consistent with the MTheory and can be expressed as sentences in first-order logic. If findings contradict the logical content of the MTheory, this can be discovered in finitely many steps. Although exact inference may not be possible for some queries, if SSBN construction will converge to the correct result if one exists.

Semantics in classical logic is typically defined in terms of possible worlds. Each possible world assigns values to random variables[8] in a manner consistent with the theory's axioms. For example, in the scenario illustrated in Figure 8, every possible world must assign value *True* to *CloakMode*(!*ST*1) and !*Z*0 to *StarshipZone*(!*ST*0), because the values of these random variables are assumed known in the scenario. The value of the random variable *ZoneNature*(!*Z*0) must be one of *DeepSpace*, *PlanetarySystems*, or *BlackHoleBoundary*, but subject to that constraint, it may have different values in different possible worlds.

In classical logic, inferences are valid if the conclusion is true in all possible worlds in which the premises are true. For example, classical logic allows us to infer that *Prev*(*Prev*(!*ST4*)) has value !*ST*2 from the information that *Prev*(!*ST*4) has value !*ST*3 and *Prev*(!*ST*3) has value !*ST*2, because the first statement is true in all possible worlds in which the latter two statements are true. But in our scenario, classical logic permits us to draw no conclusions about the value of *ZoneNature*(!*Z*0) except that it is one of the three values *DeepSpace*, *PlanetarySystems*, or *Black-HoleBoundary*.

An MTheory assigns probabilities to sets of worlds. The probability assignments ensure that the set of worlds consistent with the logical content of the MTheory has probability 100%. Each random variable instance maps a possible world to the value of the random variable in that world. In statistics, random variables are defined as functions mapping a sample space to an outcome set. For MEBN random variable instances, the sample space is the set of possible worlds. For example, *ZoneNature*(!*Z*0) maps a possible world to the nature of the zone labeled !*Z*0 in that world. The probability that !*Z*0 is a deep space zone is the total probability of the set of possible worlds for which *ZoneNature*(!*Z*0) has value *DeepSpace*.

In any given possible world, the generic random variable class *ZoneNature*(*z*) maps its argument to the nature of the zone whose identifier was substituted for the argument *z*. Thus, the sample space for the random variable class *ZoneNature*(*z*) is the set of unique identifiers that can be substituted for the argument *z*. Information about statistical regularities among zones is represented by the local distributions of the MFrags whose arguments are zones. As we saw in the section on learning, MFrags for parameter and structure learning can help us to use information about zones we have observed to make better predictions about zones we have not yet seen.

As we obtain more information about which possible world might be the actual world, we need to adjust the probabilities of all related properties of the world in a logically coherent manner. This is accomplished by adding findings to our MTheory to represent the new information, and then using Bayesian conditioning to update the probability distribution represented by the revised MTheory.

For example, suppose we learn there is at least one enemy ship in !*Z*0. This information means that worlds in which *ZoneEShips*(!*Z*0) has value *Zero* are no longer possible. In classical logic, this new information makes no difference to the inferences we can draw about *ZoneNature*(!*Z*0). All three values were possible before we learned there was an enemy ship present, and all three values remain possible. The situation is different in a probabilistic logic. To revise our probabilities, we first assign probability zero to the set of worlds in which !*Z*0 contains no enemy ships. Then, we divide the probabilities of the remaining worlds by our prior probability that *ZoneEShips*(!*Z*0) had a value other than *Zero*. This ensures that the set of worlds consistent with our new knowledge has probability 100%. These operations can be accomplished in a computationally efficient manner using SSBN construction.

Just as in classical logic, all three values of *ZoneEShips*(!*Z*0) remain possible. However, their probabilities are different from their previous values. Because deep space zones are more likely than other zones to contain no ships, more of the probability in the discarded worlds was assigned to worlds in which !*Z*0 was a deep space zone than to worlds in which !*Z*0 was not in deep space. Worlds that remain possible tended to put more probability on planetary systems and black hole boundaries than on deep space. The result is a substantial reduction in the probability that !*Z*0 is in deep space.

Achieving full first-order expressive power in a Bayesian logic is non-trivial. This requires the ability to represent an unbounded or possibly infinite number of random variables, some of which may have an unbounded or possibly infinite number of random possible values. We also need to be able to represent recursive definitions and random variables that may have an unbounded or possibly infinite

[8] In classical logic, the terms *predicate* and *function* are used in place of Boolean and non-Boolean random variables, respectively. Predicates must have value *True* or *False*, and cannot have value *Absurd*.

number of parents. Random variables taking values in uncountable sets such as the real numbers present additional difficulties. Details on how MEBN handles these subtle issues are provided by Laskey (2005).

Related Research

Hidden Markov models, or HMMs, (Baum & Petrie 1966, Elliott *et al.* 1995, Rabiner 1989) have been applied extensively in pattern recognition applications. HMMs can be viewed as a special case of dynamic Bayesian networks, or DBNs (Murphy 1998). A HMM is a DBN having hidden states with no internal structure that *d*-separate observations at different time steps. Partially dynamic Bayesian networks, also called temporal Bayesian networks (Takikawa et al. 2001) extend DBNs to include static variables. These formalisms augment standard Bayesian networks with a capability for temporal recursion.

BUGS (Buntine 1994a, Gilks *et al.* 1994, Spiegelhalter *et al.* 1996) is a software package that implements the Plates language. Plates represent repeated fragments of directed or undirected graphical models. Visually, a plate is represented as a rectangle enclosing a set of repeated nodes. Strengths of plates are the ability to handle continuous distributions without resorting to discretization, and support for parameter learning in a wide variety of parameterized statistical models. The main weakness is the lack of a direct, explicit way to represent uncertainty about model structure. There is a natural translation from plates to MFrags. See Laskey (2005) for more on the relationship between plates and MFrags.

Object-oriented Bayesian Networks (Bangsø & Wuillemin 2000, Koller & Pfeffer 1997, Langseth & Nielsen 2003) represent entities as instances of object classes with class-specific attributes and probability distributions. Probabilistic Relational Models (PRM) (Getoor et al. 2001, Getoor et al. 2000, Pfeffer 2001, Pfeffer et al. 1999) integrate the relational data model (Codd 1970) and Bayesian networks. PRMs extend standard Bayesian Networks to handle multiple entity types and relationships among them, providing a representation in which it is easy to obtain consistent probabilities over a relational database. PRMs cannot express arbitrary quantified first-order sentences and do not support recursion. Although PRMs augmented with DBNs can support limited forms of recursion, they still do not support general recursive definitions.

Finally, DAPER (Heckerman et al. 2004) combines the entity-relational model with DAG models to express probabilistic knowledge about structured entities and their relationships. Any model constructed in Plates or PRM can be represented by DAPER. Thus, DAPER is a unifying language for expressing relational probabilistic knowledge. DAPER expresses probabilistic models over finite databases, and cannot represent arbitrary FOPC expressions involving quantifiers. Therefore, like other languages discussed above, DAPER does not achieve full FOPC representational power. MEBN provides the formal mathematical support to achieve this objective, and could provide a logical foundation for extending the expressive power of any of the above formalisms.

Discussion and Future Work

MEBN logic brings together two different areas of research: probabilistic reasoning and classical logic. The ability to perform plausible reasoning with the expressiveness of Fisrt-Order Logic opens the possibility to address problems of greater complexity than heretofore possible in a wide variety of application domains.

The flexibility of the framework defined in Laskey (2005) allows it to serve as the logical basis for any typed knowledge representation. For example, Quiddity*Suite™ is a frame-based relational modeling toolkit that implements MEBN logic and is being used to address a wide range of applications ranging from visual target recognition to multi-sensor data fusion to dynamic decision systems in the C3I arena (Fung et al. 2004).

XML-based languages such as RDF and OWL are currently being developed using subsets of FOL. MEBN logic can provide a logical foundation for extensions that support plausible reasoning. As an example, we are currently developing PR-OWL, a MEBN-based extension to the semantic web language OWL. Our objective is to create a language capable of representing and reasoning with probabilistic ontologies. This technology would have many possible applications to the Semantic Web, which is an open environment where uncertainty is a rule, thus deeming the current deterministic approaches not the most suitable tool for the challenge.

Probabilistic ontologies are also a very promising technique for addressing the semantic mapping problem, a difficult task whose applications range from automatic Semantic Web agents, which must be

able to deal with multiple, diverse ontologies, to automated decision systems, which usually have to interact and reason with many legacy systems, each having its own distinct rules, assumptions, and terminologies.

MEBN is still in its infancy as a logic, but has already shown the potential to provide the necessary mathematical foundation for plausible reasoning in an open world characterized by many interacting entities related to each other in diverse ways and having many uncertain features and relationships.

Acknowledgements

Grateful acknowledgement is due to the Brazilian Air Force for supporting Paulo Costa during his PhD studies at George Mason University. Kathryn Laskey's work was partially supported under a contract with the Office of Naval Research, number N00014-04-M-0277. The authors extend thanks to Sepideh Mirza and Mehul Revankar for comments on an earlier draft, to the GMU decision theory seminar participants whose many insightful questions helped us to clarify both our thinking and our writing, and to Francis Fung, Tod Levitt, Mike Pool, and Ed Wright for many helpful discussions. Last but not least, this paper is dedicated to Danny Pearl, and to the hope that in the 24th Century, Danny's writings and his father's research will be remembered for their role in bringing about Danny's dream of a world in which all cultures and faiths live together in harmony.

References

Bangsø, O., & Wuillemin, P.-H. (2000). Object Oriented Bayesian Networks: A Framework for Topdown Specification of Large Bayesian Networks and Repetitive Structures (No. CIT-87.2-00-obphw1): Department of Computer Science, Aalborg University.

Baum, L. E., & Petrie, T. (1966). Statistical inference for probabilistic functions of finite state Markov chains. Annals of Mathematical Statistics, 37, 1554-1563.

Binford, T., Levitt, T. S., & Mann, W. B. (1987). Bayesian Inference in Model-Based Machine Vision. In a. P. C. C. T. S. Levitt (Ed.), Uncertainty in Artificial Intelligence: Proceedings of the Third Workshop. Seattle, WA.

Buntine, W. L. (1994a). Learning with Graphical Models (Technical Report No. FIA-94-03): NASA Ames Research Center, Artificial Intelligence Research Branch.

Buntine, W. L. (1994b). Operations for Learning with Graphical Models. Journal of Artificial Intelligence Research, 2, 159-225.

Charniak, E. (1991). Bayesian Networks Without Tears. AI Magazine, 12, 50-63.

Charniak, E., & Goldman, R. P. (1989a). Plan recognition in stories and in life. Paper presented at the Fifth Workshop on Uncertainty in Artificial Intelligence, Mountain View, California.

Charniak, E., & Goldman, R. P. (1989b, August 1989). A semantics for probabilistic quantifier-free first-order languages with particular application to story understanding. Paper presented at the Eleventh International Joint Conference on Artificial Intelligence, Detroit, Michigan.

Codd, E. F. (1970). A relational model for large shared data banks. Communications of the ACM, 13(6), 377-387.

Cooper, G. F., & Herskovits, E. (1992). A Bayesian Method for the Induction of Probabilistic Networks from Data. Machine Learning, 9, 309-347.

Druzdzel, M. J., & Simon, H., A. (1993). Causality in Bayesian belief networks. Paper presented at the Ninth Annual Conference on Uncertainty in Artificial Intelligence (UAI-93), San Francisco, CA.

Elliott, R. J., Aggoun, L., & Moore, J. B. (1995). Hidden Markov Models: Estimation and Control. New York, New York: Springer-Verlag.

Friedman, N., & Koller, D. (2000). Being Bayesian about network structure. Paper presented at the Sixteenth Conference on Uncertainty in Artificial Intelligence, San Mateo, California.

Fung, F., Laskey, K. B., Pool, M., Takikawa, M., & Wright, E. J. (2004). PLASMA: combining predicate logic and probability for information fusion and decision support. Paper presented at the AAAI Spring Symposium, Stanford, CA.

Getoor, L., Friedman, N., Koller, D., & Pfeffer, A. (2001). Learning Probabilistic Relational Models. New York, New York: Springer-Verlag.

Getoor, L., Koller, D., Taskar, B., & Friedman, N. (2000). Learning probabilistic relational models with structural uncertainty. Paper presented at the ICML-2000 Workshop on Attribute-Value and Relational Learning:Crossing the Boundaries, Standford, California.

Gilks, W., Thomas, A., & Spiegelhalter, D. J. (1994). A language and program for complex Bayesian modeling. The Statistician, 43, 169-178.

Hansson, O., & Mayer, A. (1989, August, 1989). Heuristic search as evidential reasoning. Paper presented at the Fifth Workshop on Uncertainty in Artificial Intelligence, Windsor, Ontario.

Heckerman, D. (1990). Probabilistic Similarity Networks. Unpublished Ph.D. Thesis, Stanford University, Stanford, CA.

Heckerman, D., Geiger, D., & Chickering, D. M. (1995a). Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. Machine Learning(20), 197-243.

Heckerman, D., Mamdani, A., & Wellman, M. P. (1995b). Real-world application of Bayesian networks. Communications of the ACM, 38(3), 24-68.

Heckerman, D., Meek, C., & Koller, D. (2004). Probabilistic models for relational data. Redmond, WA: Microsoft Corporation.

Jensen, F. V. (1996). An Introduction to Bayesian Networks. New York: Springer-Verlag.

Jensen, F. V. (2001). Bayesian Networks and Decision Graphs (2001 ed.). New York: Springer-Verlag.

Jordan, M. I., (Ed.). (1999). Learning in Graphical Models. Cambridge, MA: MIT Press.

Koller, D., & Pfeffer, A. (1997). Object-Oriented Bayesian Networks. Paper presented at the Uncertainty in Artificial Intelligence: Proceedings of the Thirteenth Conference, San Francisco, CA.

Langseth, H., & Nielsen, T. (2003). Fusion of Domain Knowledge with Data for Structured Learning in Object-Oriented Domains. Journal of Machine Learning Research.

Laskey, K. B. (2005, March 15, 2005). First-order Bayesian logic. Retrieved Mar 3, 2005, from http://ite.gmu.edu/~klaskey/publications.html

Lauritzen, S., & Spiegelhalter, D. J. (1988). Local computation and probabilities on graphical structures and their applications to expert systems. Journal of Royal Statistical Society, 50(2), 157-224.

Mahoney, S. M., & Laskey, K. B. (1998). Constructing situation specific networks. Paper presented at the Uncertainty in Artificial Intelligence: Proceedings of the Fourteenth Conference, San Mateo, CA.

Murphy, K. (1998). Dynamic Bayesian networks: representation, inference and learning. Unpublished Doctoral Dissertation, University of California, Berkeley.

Neapolitan, R. E. (1990). Probabilistic Reasoning in Expert Systems: Theory and Algorithms. New York: John Wiley and Sons, Inc.

Oliver, R. M., & Smith, J. Q. (1990). Influence Diagrams, Belief Nets and Decision Analisys (1st ed.). New York, NY: John Willey & Sons Inc.

Pearl, J. (1988). Probabilistic reasoning in intelligent systems: networks of plausible inference. San Mateo, CA: Morgan Kaufmann Publishers.

Pearl, J. (2000). Causality: models, reasoning, and inference. Cambridge, U.K.: Cambridge University Press.

Pfeffer, A. (2001). IBAL: A Probabilistic Rational Programming Language International. Paper presented at the Joint Conference on Artificial Intelligence (IJCAI).

Pfeffer, A., Koller, D., Milch, B., & Takusagawa, K. T. (1999). SPOOK: A system for probabilistic object-oriented knowledge representation. Paper presented at the Uncertainty in Artificial Intelligence: Proceedings of the Fifteenth Conference, San Mateo, CA.

Rabiner, L. R. (1989, February 1989). A tutorial on hidden Markov models and selected applications in speech recognition. Paper presented at the IEEE.

Sowa, J. F. (2000). Knowledge representation: logical, philosophical, and computational foundations. Pacific Grove: Brooks/Cole.

Spiegelhalter, D. J., Thomas, A., & Best, N. (1996). Computation on Graphical Models. Bayesian Statistics, 5, 407-425.

Stone, L. D., Barlow, C. A., & Corwin, T. L. (1999). Bayesian multiple target tracking. Boston, MA: Artech House.

Takikawa, M., d'Ambrosio, B., & Wright, E. (2002). Real-time inference with large-scale temporal Bayes nets. Paper presented at the Uncertainty in Artificial Intelligence.